
	<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: 1</p>

INFORME DE LABORATORIO

No 04

INFORMACIÓN BÁSICA					
ASIGNATURA:	Fundamentos de la Programación 2				
TÍTULO DE LA PRÁCTICA:	Arreglos y Objetos Estándar				
NÚMERO DE PRÁCTICA:	04	AÑO LECTIVO:	2024-B	NRO. SEMESTRE:	II
FECHA DE PRESENTACIÓN	18/10/2024	HORA DE PRESENTACIÓN	23:30:05		
INTEGRANTE (s) Subia Huaicane Edson Fabricio				NOTA (0-20)	Nota colocada por el docente
DOCENTE(s): Lino José Pinto Oppe					

RESULTADOS Y PRUEBAS
<p>• EJERCICIOS RESUELTOS:</p> <p>Actividad VIDEOJUEGO de SOLDADOS:</p> <ul style="list-style-type: none"> ○ Cree un Proyecto llamado Laboratorio5 ○ Usted podrá reutilizar las dos clases Nave.java y DemoBatalla.java. creadas en Laboratorio 3 y 4 ○ Inicializar el tablero con n soldados aleatorios entre 1 y 10. Cada soldado tendrá un nombre autogenerado: Soldado0, Soldado1, etc., un valor de nivel de vida autogenerado aleatoriamente [1...5], la fila y columna también autogenerados aleatoriamente (verificar que no puede haber 2 soldados en el mismo cuadrado). Se debe mostrar el tablero con todos los soldados creados (usar caracteres como _ y otros). Además, mostrar los datos del Soldado con mayor nivel de vida, el promedio de nivel de vida de todos los soldados creados, el nivel de vida de todo el ejército, los datos de todos los soldados en el orden que fueron creados y un ranking de poder de todos los soldados creados, del que tiene más nivel de vida al que tiene menos (usar al menos 2 algoritmos de ordenamiento). <p>En este enlace se encuentra mi repositorio y los commits que realicé para la creación y/o mejora de este programa: https://github.com/Q3son/Videojuego_Soldados.git</p> <p>Mis COMMITS:</p> <ul style="list-style-type: none"> • Este es el primer commit destacable que realicé, aquí creé la clase Soldado y el constructor para el buen funcionamiento del main:

Esta es la primera versión de mi clase Soldado: ↕



- Se crearon los atributos: Nombre, NivelVida, Fila y Columna, para luego ser usados en el metodo principal main

VideoJuego_Soldados\Soldado.java

```
1  + //BY: SUBIA_EDSON_FP2
2  +
1  3  public class Soldado {
4  +     private String nombre;
5  +     private int nivelVida;
6  +     private int fila;
7  +     private int columna;
8  +
9  +     public Soldado(String nombre, int nivelVida, int fila, int columna) {
10 +         this.nombre = nombre;
11 +         this.nivelVida = nivelVida;
12 +         this.fila = fila;
13 +         this.columna = columna;
14 +     }
15 +
16 +     public String getNombre() {
17 +         return nombre;
18 +     }
19 +
20 +     public int getNivelVida() {
21 +         return nivelVida;
22 +     }
23 +
24 +     public int getFila() {
25 +         return fila;
26 +     }
27 +
```

```
27 +
28 +     public int getColumna() {
29 +         return columna;
30 +     }
31
32 +     @Override
33 +     public String toString() {
34 +         return "Nombre: " + nombre + ", Nivel de vida: " + nivelVida + ", Posición: (" + fila + ", " + columna + ")";
35 +     }
36 }
```

Aquí destaco los atributos como: Nombre, NivelVida, Fila y Columna, para luego ser usados en el método principal main.

	<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: 3</p>



- Ésta es la primera versión de mi Videojuego2 de Soldados, aquí destaco la creación de métodos de ordenamiento (Burbuja y Selección), la Inicialización del Objeto Soldado y la reutilización de las bases de los laboratorios 3 y 4. //El estilo está en una versión preliminar.

1ERA VERSION Videojuego (main): 📄

- En esta versión, se han desarrollado métodos para mostrar el tablero a utilizar, la creación del Objeto Soldado, los métodos por ordenamiento Burbuja y Selección, para la buena organización y funcionamiento del programa.

```
Videojuego_Soldados:Videojuego2.java
1 + import java.util.*;
2 + //BY: SUBIA_EDSON_FP2
3 public class Videojuego2 {
4
5 +     private static final int TAMAÑO_TABLERO = 10; // Tablero de 10x10
6 +     private Soldado[][] tablero;
7 +     private Soldado[] soldados;
8 +     private int cantidadSoldados;
9 +     public static void main(String[] args) {
10 +
11 +         Videojuego2 juego = new Videojuego2(10); // Cambia la cantidad según lo necesites
12 +         juego.mostrarTablero();
13 +         System.out.println("Soldado con mayor nivel de vida: " + juego.soldadoConMayorVida());
14 +         System.out.println("Promedio de nivel de vida: " + juego.promedioNivelVida());
15 +         System.out.println("Nivel de vida total del ejército: " + juego.nivelVidaTotal());
16 +         juego.mostrarDatosSoldados();
17 +         juego.rankingDePoder();
18 +         juego.rankingPorNombre(); // Muestra ranking por nombre
19 +     }
20 +
21 +     public Videojuego2(int cantidad) {
22 +         tablero = new Soldado[TAMAÑO_TABLERO][TAMAÑO_TABLERO];
23 +         soldados = new Soldado[cantidad];
24 +         this.cantidadSoldados = cantidad;
25 +         inicializarSoldados();
26 +     }
27 +
28 +     private void inicializarSoldados() {
29 +         Random random = new Random();
30 +         int soldadosCreados = 0;
31 +
32 +         while (soldadosCreados < cantidadSoldados) {
33 +             int fila = random.nextInt(TAMAÑO_TABLERO);
34 +             int columna = random.nextInt(TAMAÑO_TABLERO);
35 +             if (tablero[fila][columna] == null) {
36 +                 String nombre = "Soldado" + soldadosCreados;
37 +                 int nivelVida = random.nextInt(5) + 1; // Nivel de vida entre 1 y 5
38 +                 Soldado soldado = new Soldado(nombre, nivelVida, fila, columna);
39 +                 tablero[fila][columna] = soldado;
40 +                 soldados[soldadosCreados] = soldado;
41 +                 soldadosCreados++;
42 +             }
43 +         }
44 +     }
45 +
46 +     public void mostrarTablero() {
47 +         System.out.println("Tablero de soldados:");
48 +         for (int i = 0; i < TAMAÑO_TABLERO; i++) {
49 +             for (int j = 0; j < TAMAÑO_TABLERO; j++) {
50 +                 if (tablero[i][j] != null) {
51 +                     System.out.print("| " + tablero[i][j].getNombre() + " ");
```

```
52 +         } else {
53 +             System.out.print("|  ");
54 +         }
55 +     }
56 +     System.out.println("|");
57 + }
58 + System.out.println();
59 + }
60 +
61 + public Soldado soldadoConMayorVida() {
62 +     Soldado maxSoldado = soldados[0];
63 +     for (int i = 1; i < cantidadSoldados; i++) {
64 +         if (soldados[i].getNivelVida() > maxSoldado.getNivelVida()) {
65 +             maxSoldado = soldados[i];
66 +         }
67 +     }
68 +     return maxSoldado;
69 + }
70 +
71 + public double promedioNivelVida() {
72 +     int suma = 0;
73 +     for (int i = 0; i < cantidadSoldados; i++) {
74 +         suma += soldados[i].getNivelVida();
75 +     }
76 +     return (double) suma / cantidadSoldados;
77 + }
78 +
79 + public int nivelVidaTotal() {
80 +     int total = 0;
81 +     for (int i = 0; i < cantidadSoldados; i++) {
82 +         total += soldados[i].getNivelVida();
83 +     }
84 +     return total;
85 + }
86 +
87 + public void mostrarDatosSoldados() {
88 +     System.out.println("Datos de todos los soldados en orden de creación:");
89 +     for (int i = 0; i < cantidadSoldados; i++) {
90 +         System.out.println(soldados[i]);
91 +     }
92 +     System.out.println();
93 + }
94 +
95 + public void rankingDePoder() {
96 +     // Ordenamiento por nivel de vida (algoritmo de ordenamiento por selección)
97 +     for (int i = 0; i < cantidadSoldados - 1; i++) {
98 +         int indexMax = i;
99 +         for (int j = i + 1; j < cantidadSoldados; j++) {
100 +             if (soldados[j].getNivelVida() > soldados[indexMax].getNivelVida()) {
101 +                 indexMax = j;
102 +             }
103 +         }
```

	<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: 5</p>

```

104 +         // Intercambiar soldados
105 +         Soldado temp = soldados[indexMax];
106 +         soldados[indexMax] = soldados[i];
107 +         soldados[i] = temp;
108 +     }
109 +
110 +     System.out.println("Ranking de poder de los soldados (ordenados por nivel de vida:");
111 +     for (int i = 0; i < cantidadSoldados; i++) {
112 +         System.out.println(soldados[i]);
113 +     }
114 +     System.out.println();
115 + }
116 +
117 + public void rankingPorNombre() {
118 +     // Ordenamiento por nombre (algoritmo de ordenamiento burbuja)
119 +     for (int i = 0; i < cantidadSoldados - 1; i++) {
120 +         for (int j = 0; j < cantidadSoldados - 1 - i; j++) {
121 +             if (soldados[j].getNombre().compareTo(soldados[j + 1].getNombre()) > 0) {
122 +                 Soldado temp = soldados[j];
123 +                 soldados[j] = soldados[j + 1];
124 +                 soldados[j + 1] = temp;
125 +             }
126 +         }
127 +     }
128 +     System.out.println("Ranking de soldados por nombre:");
129 +     for (int i = 0; i < cantidadSoldados; i++) {
130 +         System.out.println(soldados[i]);
131 +     }
132 +     System.out.println();
133 + }
134 + }

```

- **Ésta es el segundo commit más relevante y la segunda versión de Videojuego, donde hemos terminado de mejorar los métodos de Ordenamiento (Burbuja y Selección):**

Mejora de visualización del tablero e Implementación de la verificación ...

...de celdas para evitar soldados duplicados en el tablero:

- Se mantuvo el formato de visualización limpio y ordenado, con espacios reservados para celdas vacías y divisiones claras entre filas.
- Se garantiza que el número de soldados generados no exceda los límites del tablero.
- Se actualizó el método inicializarSoldados para asegurar que no se coloquen más de un soldado en la misma celda.
- Se aseguró que la lógica de creación de soldados funcione correctamente sin entrar en bucles infinitos.

```

37 + // Imprimir la cabecera del tablero
38 + System.out.println("Tablero de Soldados:");
39 +
40 + // Crear una línea divisoria que ocupe todo el ancho del tablero
41 + String lineaDivisoria = new String(new char[100]).replace("\0", "-"); // Línea divisoria de 110 caracteres
42 +
43 + // Mostrar el tablero en formato de celdas
44 + for (int fila = 0; fila < 10; fila++) {
45 + // Imprimir línea divisoria para la fila
46 + System.out.println(lineaDivisoria);
47 + System.out.print("|"); // Iniciar la fila con una barra vertical
48 +
49 + for (int columna = 0; columna < 10; columna++) {
50 + // Variable para almacenar el contenido de la celda
51 + String celda = " "; // Espacio vacío para 9 caracteres
52 +
53 + for (int i = 0; i < soldados.length; i++) {
54 + if (soldados[i] != null && soldados[i].getFila() == fila && soldados[i].getColumna() == columna) {

```

```



55 + // Si hay un soldado en esta posición, actualizamos la celda
56 + celda = String.format("%-9s", soldados[i].getNombre()); // Asignar el nombre del soldado con formato
57 + break;
58 + }
59 + }
60 +
61 + // Imprimir la celda con el contenido
62 + System.out.print(celda + "|"); // Concatenar con el símbolo de celda
63 + }
64 + System.out.println("|");
65 + // Finalizar la fila
66 + System.out.println(); // Nueva línea para la siguiente fila
67 + }
68 + System.out.println();
69 + // Imprimir la línea divisoria al final
70 + System.out.println(lineaDivisoria); // Imprimir línea divisoria final

```

- **Para la versión final agregué los llamados a los nuevos métodos en el Main y los enumeré para seguir la lógica correspondiente (los métodos y funciones más importantes) // También reduje las líneas de código menos relevantes, y omití algunas llaves, sin afectar el funcionamiento del programa**

Implementación final del juego de tablero con soldados: ✨

- Se inicializa un tablero de soldados con asignación aleatoria de atributos.
- Se implementa el método `mostrarTablero` para visualizar el estado del juego.
- Se añade el método `soldadoConMayorVida` para encontrar el soldado con el mayor nivel de vida.
- Se implementan métodos para calcular el promedio y total de nivel de vida del ejército.
- Se añaden métodos `rankingDePoder` (ordenamiento por selección) y `rankingPorNombre` (ordenamiento burbuja) para clasificar soldados.
- Se organizan los comentarios y se mejora la legibilidad del código.

	<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: 7</p>

11	+ // 1. Constructor para inicializar el juego con un número determinado de soldados
63	+ // 4. Método para encontrar el soldado con mayor nivel de vida
64	public Soldado soldadoConMayorVida() {
65	Soldado maxSoldado = soldados[0]; // Asume que hay al menos un soldado
66	for (int i = 1; i < cantidadSoldados; i++) {
	@@ -80,6 +71,7 @@ public class Videojuego2 {
71	return maxSoldado;
72	}
73	
74	+ // 5. Método para calcular el promedio de nivel de vida de los soldados
75	public double promedioNivelVida() {
76	int suma = 0;
77	for (int i = 0; i < cantidadSoldados; i++) {
	@@ -88,6 +80,7 @@ public class Videojuego2 {
80	return (double) suma / cantidadSoldados;
81	}

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)

<pre> 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 </pre>	<pre> //BY: SUBIA_EDSON_FP2 public class Soldado { private String nombre; private int nivelVida; private int fila; private int columna; public Soldado(String nombre, int nivelVida, int fila, int columna) { this.nombre = nombre; this.nivelVida = nivelVida; this.fila = fila; this.columna = columna; } public String getNombre() { return nombre; } public int getNivelVida() { return nivelVida; } public int getFila() { return fila; } public int getColumna() { return columna; } @Override public String toString() { return "Nombre: " + nombre + ", Nivel de vida: " + nivelVida + ", Posición: (" + fila + ", " + columna + ")"; } } </pre>
---	--

```
You, 37 minutes ago | 1 author (You)
1  import java.util.*;
You, 37 minutes ago | 1 author (You)
2  // BY: SUBIA_EDSON_FP2
3
4  public class Videojuego2 {
5
6      private static final int TAMAÑO_TABLERO = 10; // Tablero de 10x10
7      private Soldado[][] tablero;
8      private Soldado[] soldados;
9      private int cantidadSoldados;
10
11     // 1. Constructor para inicializar el juego con un número determinado de soldados
12     public Videojuego2(int cantidad) { // You, 4 hours ago • 1ERA VERSION Videojuego (main): ...
13         tablero = new Soldado[TAMAÑO_TABLERO][TAMAÑO_TABLERO];
14         soldados = new Soldado[cantidad];
15         this.cantidadSoldados = cantidad;
16         inicializarSoldados(); // Inicializa los soldados en el tablero
17     }
18
19     // 2. Método para crear soldados de manera aleatoria en el tablero
20     private void inicializarSoldados() {
21         Random random = new Random();
22         int soldadosCreados = 0;
23
24         while (soldadosCreados < cantidadSoldados) {
25             int fila = random.nextInt(TAMAÑO_TABLERO);
26             int columna = random.nextInt(TAMAÑO_TABLERO);
27             if (tablero[fila][columna] == null) { // Verifica que el espacio esté vacío
28                 String nombre = "Soldado" + soldadosCreados; // Nombre del soldado
29                 int nivelVida = random.nextInt(bound:5) + 1; // Nivel de vida entre 1 y 5
30                 Soldado soldado = new Soldado(nombre, nivelVida, fila, columna);
31                 tablero[fila][columna] = soldado; // Asignar el soldado al tablero
32                 soldados[soldadosCreados] = soldado; // Guardar el soldado en el array
33                 soldadosCreados++; // Incrementar el contador de soldados creados
34             }
35         }
36     }
37
38     // 3. Método para mostrar el tablero en la consola
39     public void mostrarTablero() {
40         System.out.println(x:"Tablero de Soldados:");
41         String lineaDivisoria = new String(new char[100]).replace(target:"\0", replacement:"-");
42
43         for (int fila = 0; fila < TAMAÑO_TABLERO; fila++) {
44             System.out.println(lineaDivisoria);
45             System.out.print(s:"|");
46
47             for (int columna = 0; columna < TAMAÑO_TABLERO; columna++) {
48                 String celda = " "; // Espacio vacío
49                 for (Soldado soldado : soldados) {
50                     if (soldado != null && soldado.getFila() == fila && soldado.getColumna() == columna) {
51                         celda = String.format(format:"%-9s", soldado.getNombre()); // Asignar el nombre del soldado
52                         break;
53                     }
54                 }
55                 System.out.print(celda + "|"); // Imprimir la celda
56             }
57             System.out.println(); // Nueva línea para la siguiente fila
58         }
59
60         System.out.println(lineaDivisoria); // Línea divisoria final
61     }
62
63     // 4. Método para encontrar el soldado con mayor nivel de vida
64     public Soldado soldadoConMayorVida() {
65         Soldado maxSoldado = soldados[0]; // Asume que hay al menos un soldado
66         for (int i = 1; i < cantidadSoldados; i++) {
67             if (soldados[i].getNivelVida() > maxSoldado.getNivelVida()) {
68                 maxSoldado = soldados[i];
69             }
70         }
71     }
72 }
```



```
70     }
71     return maxSoldado;
72 }
73
74 // 5. Método para calcular el promedio de nivel de vida de los soldados
75 public double promedioNivelVida() {
76     int suma = 0;
77     for (int i = 0; i < cantidadSoldados; i++) {
78         suma += soldados[i].getNivelVida();
79     }
80     return (double) suma / cantidadSoldados;
81 }
82
83 // 6. Método para calcular el nivel de vida total del ejército
84 public int nivelVidaTotal() {
85     int total = 0;
86     for (int i = 0; i < cantidadSoldados; i++) {
87         total += soldados[i].getNivelVida();
88     }
89     return total;
90 }
91
92 // 7. Método para mostrar los datos de todos los soldados
93 public void mostrarDatosSoldados() {
94     System.out.println(x:"Datos de todos los soldados en orden de creación:");
95     for (int i = 0; i < cantidadSoldados; i++) {
96         System.out.println(soldados[i]);
97     }
98     System.out.println();
99 }
100
101 // 8. Método para clasificar soldados por poder (nivel de vida)
102 public void rankingDePoder() {
103     // Ordenamiento por selección (mejorado)
104     for (int i = 0; i < cantidadSoldados - 1; i++) {
105         int indexMax = i;
106         for (int j = i + 1; j < cantidadSoldados; j++) {
107             if (soldados[j].getNivelVida() > soldados[indexMax].getNivelVida()) {
108                 indexMax = j;
109             }
110         }
111         // Solo intercambiar si es necesario
112         if (indexMax != i) {
113             Soldado temp = soldados[indexMax];
114             soldados[indexMax] = soldados[i];
115             soldados[i] = temp;
116         }
117     }
118
119     System.out.println(x:"Ranking de poder de los soldados (ordenados por nivel de vida):");
120     for (int i = 0; i < cantidadSoldados; i++) {
121         System.out.println(soldados[i]);
122     }
123     System.out.println();
124 }
125
126 // 9. Método para clasificar soldados por nombre
127 public void rankingPorNombre() {
128     // Ordenamiento burbuja (mejorado)
129     boolean Cambiazo;
130     for (int i = 0; i < cantidadSoldados - 1; i++) {
131         Cambiazo = false;
132         for (int j = 0; j < cantidadSoldados - 1 - i; j++) {
133             if (soldados[j].getNombre().compareTo(soldados[j + 1].getNombre()) > 0) {
134                 // Intercambio
135                 Soldado temp = soldados[j];
136                 soldados[j] = soldados[j + 1];
137                 soldados[j + 1] = temp;
138                 Cambiazo = true;
139             }
140         }
141     }
142 }
```


Soldado con mayor nivel de vida: Nombre: Soldado5, Nivel de vida: 5, Posición: (5, 3)

Promedio de nivel de vida: 3.1

Nivel de vida total del ejército: 31

Datos de todos los soldados en orden de creación:

Nombre: Soldado0, Nivel de vida: 3, Posición: (4, 5)

Nombre: Soldado1, Nivel de vida: 3, Posición: (9, 9)

Nombre: Soldado2, Nivel de vida: 1, Posición: (1, 1)

Nombre: Soldado3, Nivel de vida: 2, Posición: (4, 7)

Nombre: Soldado4, Nivel de vida: 3, Posición: (6, 8)

Nombre: Soldado5, Nivel de vida: 5, Posición: (5, 3)

Nombre: Soldado6, Nivel de vida: 5, Posición: (0, 7)

Nombre: Soldado7, Nivel de vida: 4, Posición: (5, 8)

Nombre: Soldado8, Nivel de vida: 4, Posición: (6, 2)

Nombre: Soldado9, Nivel de vida: 1, Posición: (4, 9)

Ranking de poder de los soldados (ordenados por nivel de vida):

Nombre: Soldado5, Nivel de vida: 5, Posición: (5, 3)

Nombre: Soldado6, Nivel de vida: 5, Posición: (0, 7)

Nombre: Soldado7, Nivel de vida: 4, Posición: (5, 8)

Nombre: Soldado8, Nivel de vida: 4, Posición: (6, 2)

Nombre: Soldado4, Nivel de vida: 3, Posición: (6, 8)

Nombre: Soldado0, Nivel de vida: 3, Posición: (4, 5)

Nombre: Soldado1, Nivel de vida: 3, Posición: (9, 9)

Nombre: Soldado3, Nivel de vida: 2, Posición: (4, 7)

Nombre: Soldado2, Nivel de vida: 1, Posición: (1, 1)

Nombre: Soldado9, Nivel de vida: 1, Posición: (4, 9)

Ranking de soldados por nombre:

Nombre: Soldado0, Nivel de vida: 3, Posición: (4, 5)

Nombre: Soldado1, Nivel de vida: 3, Posición: (9, 9)

Nombre: Soldado2, Nivel de vida: 1, Posición: (1, 1)

Nombre: Soldado3, Nivel de vida: 2, Posición: (4, 7)

Nombre: Soldado4, Nivel de vida: 3, Posición: (6, 8)



Nombre: Soldado5, Nivel de vida: 5, Posición: (5, 3)

Nombre: Soldado6, Nivel de vida: 5, Posición: (0, 7)

Nombre: Soldado7, Nivel de vida: 4, Posición: (5, 8)

Nombre: Soldado8, Nivel de vida: 4, Posición: (6, 2)

Nombre: Soldado9, Nivel de vida: 1, Posición: (4, 9)

	<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: 12</p>

¿Con qué valores comprobaste que tu práctica estuviera correcta?

Comprobé la práctica utilizando diversas combinaciones de entradas al crear los soldados en el juego. Ingresé nombres de soldados como "Soldado1", "Soldado2" y "Soldado3", variando los valores para la fila (por ejemplo, 1, 2, 3), columna (0, 1, 2) y nivel de vida (10, 20, 30). Para probar los nuevos métodos, utilicé nombres existentes y no existentes al buscar soldados por nombre, así como diferentes límites de nivel de vida para las búsquedas de soldados según sus puntajes. Además, utilicé métodos de ordenamiento como burbuja y selección en soldados con diversos valores de nivel de vida aleatorios entre 1 y 5, para evaluar su eficacia.

¿Qué resultado esperabas obtener para cada valor de entrada?



Esperaba que el juego tuviera el siguiente comportamiento:

- Al ordenar soldados por nivel de vida (método de selección): Se esperaba que los soldados se organizaran correctamente de acuerdo con sus niveles de vida, reflejando el orden ascendente esperado después de aplicar cada método.
- Al buscar soldados por nombre (método Burbuja): El programa encontraría correctamente los soldados que coincidieran con el nombre autogenerado.
- Al mostrar el soldado con mayor nivel de vida: Se esperaba que se identificara correctamente el soldado con el mayor nivel de vida en el arreglo.

¿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 se mostraron correctamente ordenados en función de sus niveles de vida, validando así la funcionalidad de los métodos de ordenamiento (algoritmo de selección) utilizados.
- Al buscar soldados por nombre (Burbuja): El programa ordenó satisfactoriamente los nombres de los soldados autogenerados por el programa
- Al mostrar el soldado con mayor nivel de vida: El programa identificó correctamente el soldado con el mayor nivel de vida en el arreglo.

	<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: 13</p>

1. CUESTIONARIO:

Los datos más importantes que consideré para realizar el código del juego de Soldados, especialmente en relación con los nuevos métodos de ordenamiento, fueron los siguientes:



1. **Soldado:** *Cada soldado tiene atributos clave como nombre, fila, columna y nivel de vida. Estos atributos son esenciales para definir las características de cada soldado dentro del tablero de juego, y son los criterios principales para los métodos de ordenamiento que hemos implementado. El nivel de vida, en particular, es un valor central en la estrategia del juego, ya que determina la resistencia de cada soldado.*
2. **Arreglo bidimensional de soldados:** *El uso de un arreglo bidimensional para almacenar los objetos de tipo Soldado es clave. Este arreglo no solo simula el tablero de juego, donde los soldados están ubicados en posiciones específicas, sino que también permite gestionar eficientemente la interacción de los soldados durante la partida. Además, facilita la aplicación de algoritmos de ordenamiento cuando se necesita reorganizar los soldados según sus niveles de vida o nombres.*
3. **Métodos de ordenamiento:** *La implementación de los métodos de ordenamiento es crucial para organizar a los soldados según diferentes criterios, como el nivel de vida o el nombre. Esto permite estructurar mejor el juego y generar dinámicas estratégicas:*
 - **Ordenamiento por selección:** *Este método se utilizó para ordenar a los soldados por nivel de vida, de menor a mayor. Nos permite identificar qué soldados tienen menor resistencia y cuáles están en mejor condición física, lo que es útil para decidir las mejores estrategias en la partida.*
 - **Ordenamiento por burbuja:** *Este algoritmo se utilizó para ordenar a los soldados por nombre, permitiendo organizar al ejército alfabéticamente. Esto mejora la búsqueda de soldados y añade un nivel de gestión en la interfaz del juego.*
4. **Interacción con el usuario:** *Los métodos de ordenamiento mejoran la experiencia del usuario al permitirle ver los soldados organizados según sus criterios preferidos, ya sea por nivel de vida o nombre. Esto agrega una dimensión de estrategia y análisis, ya que los jugadores pueden evaluar fácilmente la capacidad de su ejército y tomar decisiones más informadas.*
5. **Comparación y actualización:** *Los métodos de comparación que implementamos para los algoritmos de selección y burbuja son esenciales para la lógica del juego. Permiten determinar cómo deben organizarse los soldados en función de su nivel de vida o nombre, y facilitan la identificación de los soldados más fuertes o las mejores posiciones estratégicas.*

CONCLUSIONES

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

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, mientras que el ordenamiento por **nombre** mediante **burbuja** facilita la gestión alfabética de los soldados, mejorando la experiencia de búsqueda y análisis.

Los resultados obtenidos durante la ejecución del juego confirmaron que los algoritmos de ordenamiento y los métodos de comparación respondieron eficazmente a las entradas del usuario, permitiendo una gestión más

	<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: 14</p>

dinámica del tablero. Además, la visualización del soldado con mayor nivel de vida y el promedio general del ejército añadieron capas de estrategia útiles para la toma de decisiones en el juego.

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

Colocar 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 LABORA](https://github.com/LINOPINTO2023/FundProq2/blob/main/entregaLaboratorio01/Hilacondo%20Emanuel%20LABORATORIO%2001.pdf)

[TORIO 01.pdf](https://github.com/Q3son/Videojuego%20Soldados.git)
<https://github.com/Q3son/Videojuego Soldados.git>

	<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: 15

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	