


	<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: 1</p>

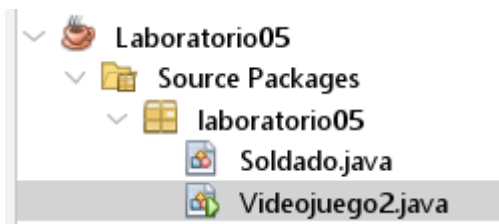
INFORME DE LABORATORIO

(formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	LABORATORIO DE FUNDAMENTOS DE PROGRAMACIÓN 2				
TÍTULO DE LA PRÁCTICA:	ARREGLOS BIDIMENSIONALES DE OBJETOS				
NÚMERO DE PRÁCTICA:	05	AÑO LECTIVO:	2024	NRO. SEMESTRE:	II
FECHA DE PRESENTACIÓN	18/10/2024	HORA DE PRESENTACIÓN	17:54		
INTEGRANTE (s) Diego Aristides Cervantes Apaza				NOTA (0-20)	
DOCENTE(s): Jose Lino Pinto Oppe					

RESULTADOS Y PRUEBAS
<p>I. EJERCICIOS RESUELTOS:</p> <ol style="list-style-type: none"> 1. Cree un Proyecto llamado Laboratorio5 2. Usted deberá crear las dos clases Soldado.java y VideoJuego2.java. Puede reutilizar lo desarrollado en Laboratorio 3 y 4. 3. Del Soldado nos importa el nombre, nivel de vida, fila y columna (posición en el tablero). 4. El juego se desarrollará en el mismo tablero de los laboratorios anteriores. Pero ahora el tablero debe ser un arreglo bidimensional de objetos. 5. 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 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: 2



CÓDIGO:



```

1 //Autor: Diego Aristides Cervantes Apaza
2 //Problema: 1. Cree un Proyecto llamado Laboratorio5
3 //2. Usted deberá crear las dos clases Soldado.java y VideoJuego2.java. Puede reutilizar lo
4 //desarrollado en Laboratorio 3 y 4.
5 //3. Del Soldado nos importa el nombre, nivel de vida, fila y columna (posición en el tablero).
6 //4. El juego se desarrollará en el mismo tablero de los laboratorios anteriores. Pero ahora el
7 //tablero debe ser un arreglo bidimensional de objetos.
8 //5. Inicializar el tablero con n soldados aleatorios entre 1 y 10. Cada soldado tendrá un nombre
9 //autogenerado: Soldado0, Soldado1, etc., un valor de nivel de vida autogenerado
10 //aleatoriamente [1..5], la fila y columna también autogenerados aleatoriamente (verificar que
11 //no puede haber 2 soldados en el mismo cuadrado). Se debe mostrar el tablero con todos los
12 //soldados creados (usar caracteres como | _ y otros). Además, mostrar los datos del Soldado
13 //con mayor nivel de vida, el promedio de nivel de vida de todos los soldados creados, el nivel
14 //de vida de todo el ejército, los datos de todos los soldados en el orden que fueron creados y
15 //un ranking de poder de todos los soldados creados, del que tiene más nivel de vida al que
16 //tiene menos (usar al menos 2 algoritmos de ordenamiento).
17 package laboratorio05;
18
19 import java.util.Random;
```

Aleatorización de números para filas, columnas y cantidad de soldados, además de la creación del arreglo bidimensional:

```

24
25 //Generación de filas y columnas aleatorias
26 int filas = rand.nextInt(8) + 3;
27 int columnas = rand.nextInt(8) + 3;
28
29
30 //Creación de arreglo bidimensional
31 Soldado[][] tablero = new Soldado[filas][columnas];
32
33 //Generación de cantidad de soldados totales
34 int numeroSoldados = rand.nextInt(10) + 1;
35
```

	<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>

Verificación de soldados no duplicados e impresión del tablero:

```

36 //Inicializar soldados aleatorios asegurando que no haya duplicados
37 for (int i = 0; i < numeroSoldados; i++) {
38     int fila, columna;
39     do {
40         fila = rand.nextInt(filas); //Filas aleatorias dentro del rango
41         columna = rand.nextInt(columnas); //Columnas aleatorias dentro del rango
42     } while (tablero[fila][columna] != null); //Aseguramos que no haya dos soldados en el mismo lugar
43
44     //Creación de nombres de soldados autogenerados
45     String nombre = "Soldado" + i;
46     int nivelDeVida = rand.nextInt(5) + 1; //Nivel de vida aleatorio
47     Soldado soldado = new Soldado();
48     soldado.setNombre(nombre);
49     soldado.setFila(fila);
50     soldado.setColumna(columna);
51     soldado.setNivelDeVida(nivelDeVida);
52     tablero[fila][columna] = soldado; //Meter un soldado generado al tablero
53 }
54
55 //Impresión de tablero
56 System.out.println("Tablero:");
57 mostrarTablero(tablero);

```

Datos del soldado con más vida, promedio de vida, vida total del ejército, impresión del soldado en el orden en el que fueron creados, ranking de soldados por nivel de vida:

```

59 //Impresión de soldado con más vida
60 Soldado soldadoMayorVida = soldadoConMasNivelDeVida(tablero);
61 System.out.println("\nSoldado con mayor nivel de vida:");
62 System.out.println(soldadoMayorVida.getNombre() + " - Nivel de vida: " + soldadoMayorVida.getNivelDeVida());
63
64 //Impresión del nivel promedio de vida de los soldados
65 double promedioNivelDeVida = promedioNivelDeVida(tablero);
66 System.out.println("Promedio de nivel de vida: " + promedioNivelDeVida);
67
68 //Impresión vida total
69 int nivelDeVidaTotal = nivelDeVidaTotal(tablero);
70 System.out.println("Nivel de vida total del ejército: " + nivelDeVidaTotal);
71
72 //Impresión de los soldados en el orden que fueron creados
73 System.out.println("\nSoldados en el orden que fueron creados:");
74 mostrarSoldadosEnOrden(tablero);
75
76 //Mostrar ranking de soldados por nivel de vida (usando burbuja)
77 Soldado[] soldadosRanking = obtenerListaDeSoldados(tablero);
78 ordenarSoldadosPorNivelDeVidaBurbuja(soldadosRanking);
79 System.out.println("\nRanking de soldados por nivel de vida (Burbuja):");
80 mostrarSoldados(soldadosRanking);
81
82 //Mostrar ranking de soldados por nivel de vida (usando selección)
83 ordenarSoldadosPorNivelDeVidaSeleccion(soldadosRanking);
84 System.out.println("\nRanking de soldados por nivel de vida (Selección):");
85 mostrarSoldados(soldadosRanking);

```

Creación de un tablero autorregulable según parámetros aleatorizados previamente:

```
88 //Método para mostrar el tablero con los soldados
89 public static void mostrarTablero(Soldado[][] tablero) {
90     // Imprimir la primera fila de letras (A, B, C, ...)
91     System.out.print(" "); // Espacio para la columna de números
92     for (int j = 0; j < tablero[0].length; j++) {
93         System.out.print(" " + (char) ('A' + j) + " ");
94     }
95     System.out.println();
96
97     //Impresión del tablero con los números en la primera columna
98     for (int i = 0; i < tablero.length; i++) {
99         System.out.print((i + 1) + " "); // Número de fila
100        if (i + 1 < 10) {
101            System.out.print(" "); // Alineación para números de una cifra
102        }
103
104        for (int j = 0; j < tablero[i].length; j++) {
105            if (tablero[i][j] == null) {
106                System.out.print("| _ ");
107            } else {
108                System.out.print("| S ");
109            }
110        }
111        System.out.println("|");
112    }
113 }
```

Definición del método para hallar el soldado con más nivel de vida:

```
115 //Método que devuelve el soldado con mayor nivel de vida
116 public static Soldado soldadoConMasNivelDeVida(Soldado[][] tablero) {
117     Soldado soldadoMayor = null;
118     for (int i = 0; i < tablero.length; i++) {
119         for (int j = 0; j < tablero[i].length; j++) {
120             Soldado actual = tablero[i][j];
121             if (actual != null && (soldadoMayor == null || actual.getNivelDeVida() > soldadoMayor.getNivelDeVida())) {
122                 soldadoMayor = actual;
123             }
124         }
125     }
126     return soldadoMayor;
127 }
```

Definición del método para hallar el promedio de vida del ejército:

```
129 //Método que calcula el promedio de nivel de vida del ejército
130 public static double promedioNivelDeVida(Soldado[][] tablero) {
131     int suma = 0;
132     int contador = 0;
133     for (int i = 0; i < tablero.length; i++) {
134         for (int j = 0; j < tablero[i].length; j++) {
135             if (tablero[i][j] != null) {
136                 suma += tablero[i][j].getNivelDeVida();
137                 contador++;
138             }
139         }
140     }
141     return (double) suma / contador;
142 }
143
```

Definición del método para hallar el nivel de vida total del ejército:

```
144 //Método que calcula el nivel de vida total del ejército
145 public static int nivelDeVidaTotal(Soldado[][] tablero) {
146     int suma = 0;
147     for (int i = 0; i < tablero.length; i++) {
148         for (int j = 0; j < tablero[i].length; j++) {
149             if (tablero[i][j] != null) {
150                 suma += tablero[i][j].getNivelDeVida();
151             }
152         }
153     }
154     return suma;
155 }
156
```

Registro de creación de los soldados:

```
157 //Método para mostrar los soldados en el orden en que fueron creados
158 public static void mostrarSoldadosEnOrden(Soldado[][] tablero) {
159     for (int i = 0; i < tablero.length; i++) {
160         for (int j = 0; j < tablero[i].length; j++) {
161             if (tablero[i][j] != null) {
162                 Soldado soldado = tablero[i][j];
163                 System.out.println(soldado.getNombre() + " | Fila: " + soldado.getFila() + " | Columna: " + soldado.getColumna() +
164                                     " | Nivel de vida: " + soldado.getNivelDeVida());
165             }
166         }
167     }
168 }
```

Registro de todos los soldados:

```
170 //Método para convertir el tablero en una lista unidimensional de soldados
171 public static Soldado[] obtenerListaDeSoldados(Soldado[][] tablero) {
172     Soldado[] soldados = new Soldado[contarSoldados(tablero)];
173     int index = 0;
174     for (int i = 0; i < tablero.length; i++) {
175         for (int j = 0; j < tablero[i].length; j++) {
176             if (tablero[i][j] != null) {
177                 soldados[index++] = tablero[i][j];
178             }
179         }
180     }
181     return soldados;
182 }
```

Definición del método que cuenta soldados:

```
184 //Método que cuenta el número de soldados en el tablero
185 public static int contarSoldados(Soldado[][] tablero) {
186     int contador = 0;
187     for (int i = 0; i < tablero.length; i++) {
188         for (int j = 0; j < tablero[i].length; j++) {
189             if (tablero[i][j] != null) {
190                 contador++;
191             }
192         }
193     }
194     return contador;
195 }
```

Definición del método de ordenación o de búsqueda burbuja:

```
197 //Método de ordenamiento por burbuja
198 public static void ordenarSoldadosPorNivelDeVidaBurbuja(Soldado[] soldados) {
199     boolean ordenado;
200     do {
201         ordenado = true;
202         for (int i = 0; i < soldados.length - 1; i++) {
203             if (soldados[i].getNivelDeVida() < soldados[i + 1].getNivelDeVida()) {
204                 Soldado temp = soldados[i];
205                 soldados[i] = soldados[i + 1];
206                 soldados[i + 1] = temp;
207                 ordenado = false;
208             }
209         }
210     } while (!ordenado);
211 }
```

Definición del método de ordenación por selección:

```
213 //Método de ordenamiento por selección
214 public static void ordenarSoldadosPorNivelDeVidaSeleccion(Soldado[] soldados) {
215     for (int i = 0; i < soldados.length - 1; i++) {
216         int maxIdx = i;
217         for (int j = i + 1; j < soldados.length; j++) {
218             if (soldados[j].getNivelDeVida() > soldados[maxIdx].getNivelDeVida()) {
219                 maxIdx = j;
220             }
221         }
222         Soldado temp = soldados[maxIdx];
223         soldados[maxIdx] = soldados[i];
224         soldados[i] = temp;
225     }
226 }
```

Definición del método que imprime una lista de soldados:

```
228 //Método para mostrar una lista de soldados
229 public static void mostrarSoldados(Soldado[] soldados) {
230     for (Soldado soldado : soldados) {
231         System.out.println(soldado.getNombre() + " | Nivel de vida: " + soldado.getNivelDeVida());
232     }
233 }
234 }
```

Clase referenciada Soldado con sus atributos y métodos:

```
19 public class Soldado {
20     private String nombre;
21     private int fila;
22     private int columna;
23     private int nivelDeVida;
24
25     // Métodos mutadores
26     public void setNombre(String n) {
27         nombre = n;
28     }
29     public void setFila(int f) {
30         fila = f;
31     }
32     public void setColumna(int c) {
33         columna = c;
34     }
35     public void setNivelDeVida(int f) {
36         nivelDeVida = f;
37     }
38
39     // Métodos accesorios
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 getNivelDeVida() {
50         return nivelDeVida;
51     }
52 }
```


II. PRUEBAS

Generación del tablero autorregulable:

Output - Laboratorio05 (run) ×

run:
Tablero:



	A	B	C	D	E
1	_	_	_	_	_
2	_	_	S	_	_
3	_	_	_	_	S
4	S	S	_	_	S
5	_	S	_	_	S

Solicitud de datos específicos necesarios para la resolución del problema:

```
Soldado con mayor nivel de vida:  
Soldado0 - Nivel de vida: 5  
Promedio de nivel de vida: 3.142857142857143  
Nivel de vida total del ejército: 22
```

Muestra del orden específico de creación de los soldados:

```
Soldados en el orden que fueron creados:  
Soldado5 | Fila: 1 | Columna: 2 | Nivel de vida: 3  
Soldado3 | Fila: 2 | Columna: 4 | Nivel de vida: 1  
Soldado0 | Fila: 3 | Columna: 0 | Nivel de vida: 5  
Soldado6 | Fila: 3 | Columna: 1 | Nivel de vida: 2  
Soldado4 | Fila: 3 | Columna: 4 | Nivel de vida: 4  
Soldado1 | Fila: 4 | Columna: 1 | Nivel de vida: 4  
Soldado2 | Fila: 4 | Columna: 4 | Nivel de vida: 3
```

	<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 style="text-align: center;">Código: GUIA-PRLE-001</p>	<p style="text-align: right;">Página: 10</p>

Ordenamiento por búsqueda burbuja de los soldados enlistado, según su nivel de vida:

```

Ranking de soldados por nivel de vida (Burbuja):
Soldado0 | Nivel de vida: 5
Soldado4 | Nivel de vida: 4
Soldado1 | Nivel de vida: 4
Soldado5 | Nivel de vida: 3
Soldado2 | Nivel de vida: 3
Soldado6 | Nivel de vida: 2
Soldado3 | Nivel de vida: 1

```

Ordenamiento por selección de los soldados de manera enlistada, según nivel de vida:

```

Ranking de soldados por nivel de vida (Selección):
Soldado0 | Nivel de vida: 5
Soldado4 | Nivel de vida: 4
Soldado1 | Nivel de vida: 4
Soldado5 | Nivel de vida: 3
Soldado2 | Nivel de vida: 3
Soldado6 | Nivel de vida: 2
Soldado3 | Nivel de vida: 1
BUILD SUCCESSFUL (total time: 0 seconds)

```

Ejecución de un segundo intento con valores totalmente aleatorios.



```
Output - Laboratorio05 (run) ×
run:
Tablero:
      A  B  C  D  E
1 | _ | _ | _ | _ |
2 | _ | _ | _ | S |
3 | _ | _ | _ | _ |
4 | _ | _ | _ | _ |
5 | _ | _ | _ | _ |
6 | _ | _ | S | _ |
7 | _ | _ | _ | _ |

Soldado con mayor nivel de vida:
Soldado0 - Nivel de vida: 3
Promedio de nivel de vida: 3.0
Nivel de vida total del ejército: 6

Soldados en el orden que fueron creados:
Soldado0 | Fila: 1 | Columna: 3 | Nivel de vida: 3
Soldado1 | Fila: 5 | Columna: 2 | Nivel de vida: 3

Ranking de soldados por nivel de vida (Burbuja):
Soldado0 | Nivel de vida: 3
Soldado1 | Nivel de vida: 3

Ranking de soldados por nivel de vida (Selección):
Soldado0 | Nivel de vida: 3
Soldado1 | Nivel de vida: 3
BUILD SUCCESSFUL (total time: 0 seconds)
```

	<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: 12

III. CUESTIONARIO:

Se ha puesto en evidencia los avances del laboratorio 05, mediante el link de github.
<https://github.com/Aristides-1/AVANCESLABSDIEGOCERVANTES>

CONCLUSIONES



Se ha trabajado de acorde a la creación de clases, atributos y métodos según el requerimiento del laboratorio, luego de haber recorrido varias propuestas. Se recalca también el uso de arreglos bidimensionales para una ejecución adrede para la creación de recuadros autorregulables.

METODOLOGÍA DE TRABAJO

Se colocó los avances en github y posteriormente los nuevos commits para el rediseño y constante mejora del código para hallar finalmente un código limpio y funcional.
 Se ha subido las pruebas en el link del repositorio del estudiante autor.

REFERENCIAS Y BIBLIOGRAFÍA

RUBRICA DEL ESTUDIANTE:

	<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: 13

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. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4		0	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	x		
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	x		
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	x		
6. Fechas	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	x		
7. Ortografía	El documento no muestra errores ortográficos.	2	x		
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		2	
TOTAL		20	14		