
	<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

(formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	Fundamentos de la Programación 2 – Laboratorio				
TÍTULO DE LA PRÁCTICA:	Arreglos Bidimensionales de Objetos				
NÚMERO DE PRÁCTICA:	05	AÑO LECTIVO:	2024	NRO. SEMESTRE:	02
FECHA DE PRESENTACIÓN	18/10/2024	HORA DE PRESENTACIÓN	17:34:32		
INTEGRANTE (s) Sergio Emilio Estrada Arce				NOTA (0-20)	Nota colocada por el docente
DOCENTE(s): Ing. Lino José Pinto Oppe					

RESULTADOS Y PRUEBAS
<p>I. EJERCICIOS RESUELTOS:</p> <p><i>Ejercicio 1:</i></p> <pre> 1 //Laboratorio Nro5 - Ejercicio 1 2 //Autor: Sergio Estrada Arce 3 //Tiempo: 2h 30h 4 package Ejercicio; 5 public class Ejercicio1 { 6 public static void main(String[] args) { 7 VideoJuego2 juego = new VideoJuego2(10, 10); 8 9 System.out.println("Tablero Inicial:"); 10 juego.mostrarTablero(); 11 12 // Ordenamiento Burbuja 13 juego.ordenarPorVidaBurbuja(); 14 System.out.println("\nRanking después de Burbuja:"); 15 juego.mostrarRanking(); 16 17 // Ordenamiento por Inserción Binaria 18 juego.ordenarPorVidaInsercionBinaria(); 19 System.out.println("\nRanking después de Inserción Binaria:"); 20 juego.mostrarRanking(); 21 } 22 }</pre>

```
1 package Ejercicio;
2 public class Soldado {
3     private String nombre;
4     private int vida;
5     private int fila;
6     private int columna;
7     // Constructor
8     public Soldado(String nombre, int vida, int fila, int columna) {
9         this.nombre = nombre;
10        this.vida = vida;
11        this.fila = fila;
12        this.columna = columna;
13    }
14    // Getters
15    public String getNombre() {
16        return nombre;
17    }
18
19    public int getVida() {
20        return vida;
21    }
22
23    public int getFila() {
24        return fila;
25    }
26
27    public int getColumna() {
28        return columna;
29    }
30    // Setters
31    public void setVida(int vida) {
32        this.vida = vida;
33    }
34    // Método toString para mostrar la información del soldado
35    @Override
36    public String toString() {
37        return "Soldado{" +
38            "nombre='" + nombre + '\'' +
39            ", vida=" + vida +
40            ", posición=(" + fila + ", " + columna + ")" +
41            '}'
42    }
43 }
```



```
1 package Ejercicio;
2 import java.util.*;
3 public class VideoJuego2 {
4     private Soldado[][] tablero;
5     private ArrayList<Soldado> listaSoldados;
6     // Constructor
7     public VideoJuego2(int filas, int columnas) {
8         tablero = new Soldado[filas][columnas];
9         listaSoldados = new ArrayList<>();
10        inicializarSoldados();
11    }
12    // Inicializar soldados con cantidad aleatoria
13    private void inicializarSoldados() {
14        Random rand = new Random();
15        int numSoldados = rand.nextInt(10) + 1;
16        for (int i=0; i<numSoldados;i++) {
17            String nombre = "Soldado" + i;
18            int nivelVida = rand.nextInt(5) + 1;
19            int fila, columna;
20            do {
21                fila = rand.nextInt(tablero.length);
22                columna = rand.nextInt(tablero[0].length);
23            } while (tablero[fila][columna] != null);
24            Soldado nuevoSoldado = new Soldado(nombre, nivelVida, fila, columna);
25            tablero[fila][columna] = nuevoSoldado;
26            listaSoldados.add(nuevoSoldado);
27        }
28    }
29    public void mostrarTablero() {
30        // Títulos de columnas (A, B, C...)
31        System.out.print(" ");
32        for (int j = 0; j < tablero[0].length; j++) {
33            System.out.print(" \s " + (char) ('A' + j) + " ");
34        }
35        System.out.println();
36        // Recorrer filas y columnas del tablero
37        for (int i = 0; i < tablero.length; i++) {
38            System.out.print("\n" + (i + 1) + " ");
39            if (i + 1 < 10) System.out.print(" "); // Alinear los números de fila
40            for (int j = 0; j < tablero[i].length; j++) {
41                if (tablero[i][j] != null) {
42                    // Mostrar el nombre del soldado en la celda
43                    String soldado = tablero[i][j].getNombre(); // Directamente desde el tablero
```

```

44         soldado = String.format("%-8s", soldado);    // Ajustar a 8 espacios
45         System.out.print("| " + soldado);
46     } else {
47         System.out.print("| _____");
48     }
49 }
50 System.out.println("|");
51 }
52 }
53 // Algoritmo de ordenamiento Burbuja (por nivel de vida)
54 public void ordenarPorVidaBurbuja() {
55     int n = listaSoldados.size();
56     for (int i = 0; i < n - 1; i++) {
57         for (int j = 0; j < n - i - 1; j++) {
58             // Cambiar la comparación para ordenar de mayor a menor
59             if (listaSoldados.get(j).getVida() < listaSoldados.get(j + 1).getVida()) {
60                 // Intercambiar
61                 Soldado temp = listaSoldados.get(j);
62                 listaSoldados.set(j, listaSoldados.get(j + 1));
63                 listaSoldados.set(j + 1, temp);
64             }
65         }
66     }
67 }
68 // Algoritmo de ordenamiento por inserción binaria (de mayor a menor)
69 public void ordenarPorVidaInsercionBinaria() {
70     for (int i = 1; i < listaSoldados.size(); i++) {
71         Soldado key = listaSoldados.get(i);
72         int pos = busquedaBinaria(listaSoldados, key, 0, i - 1);
73         for (int j = i - 1; j >= pos; j--) {
74             listaSoldados.set(j + 1, listaSoldados.get(j));
75         }
76         listaSoldados.set(pos, key);
77     }
78 }
79 // Búsqueda binaria para encontrar la posición de inserción
80 private int busquedaBinaria(List<Soldado> lista, Soldado key, int low, int high) {
81     while (low <= high) {
82         int mid = (low + high) / 2;
83         // Cambiar la comparación para ordenar de mayor a menor
84         if (key.getVida() > lista.get(mid).getVida()) {
85             high = mid - 1;
86         } else {
87             low = mid + 1;
88         }
89     }
90     return low;
91 }
92 // Mostrar el ranking de soldados por nivel de vida
93 public void mostrarRanking() {
94     System.out.println("Ranking de Soldados (por nivel de vida):");
95     for (Soldado soldado : listaSoldados) {
96         System.out.println(soldado.getNombre() + " - Vida: " + soldado.getVida());
97     }
98 }
99 }

```

[illegible]

	<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: 6</p>

Ranking después de Burbuja:

Ranking de Soldados (por nivel de vida):

Soldado1 - Vida: 5

Soldado6 - Vida: 5

Soldado8 - Vida: 5

Soldado5 - Vida: 4

Soldado3 - Vida: 3

Soldado4 - Vida: 3

Soldado0 - Vida: 2

Soldado7 - Vida: 2

Soldado2 - Vida: 1

Ranking después de Inserción Binaria:

Ranking de Soldados (por nivel de vida):

Soldado1 - Vida: 5

Soldado6 - Vida: 5

Soldado8 - Vida: 5

Soldado5 - Vida: 4

Soldado3 - Vida: 3

Soldado4 - Vida: 3

Soldado0 - Vida: 2

Soldado7 - Vida: 2

Soldado2 - Vida: 1

Tablero Inicial:

	A	B	C	D	E	F	G	H	I	J
1			Soldado1							
2										
3										
4										
5										
6										
7			Soldado0							
8										
9										
10										

Ranking después de Burbuja:
Ranking de Soldados (por nivel de vida):

Soldado0 - Vida: 3

Soldado1 - Vida: 1

Ranking después de Inserción Binaria:
Ranking de Soldados (por nivel de vida):



Soldado0 - Vida: 3

Soldado1 - Vida: 1

<terminated> Ejercicio1 (1) [Java Application] C:\Users\Usuario\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.11.v2022-03-01\jre\bin\java.exe

Tablero Inicial:

	A	B	C	D	E	F	G	H	I	J
1										
2										
3				Soldado7						Soldado4
4										
5				Soldado1		Soldado2				
6		Soldado8								
7				Soldado0	Soldado6					
8	Soldado5									
9										
10						Soldado3				

	<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: 8</p>

Ranking después de Burbuja:

Ranking de Soldados (por nivel de vida):

Soldado1 - Vida: 5

Soldado6 - Vida: 5

Soldado3 - Vida: 4

Soldado7 - Vida: 4

Soldado8 - Vida: 4

Soldado4 - Vida: 3

Soldado5 - Vida: 2

Soldado0 - Vida: 1

Soldado2 - Vida: 1

Ranking después de Inserción Binaria:

Ranking de Soldados (por nivel de vida):

Soldado1 - Vida: 5

Soldado6 - Vida: 5

Soldado3 - Vida: 4

Soldado7 - Vida: 4

Soldado8 - Vida: 4



Soldado4 - Vida: 3

Soldado5 - Vida: 2

Soldado0 - Vida: 1

Soldado2 - Vida: 1

Commits:

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

Commits

main		All users	All time
Commits on Oct 18, 2024			
Creación de la clase VideoJuego2. Inicialización del tablero y lista de soldados, visualización del tablero, y algoritmos de ordenamiento por vida (burbuja e inserción binaria). Se incluye también ...		03e055e	<>
SrEstrada committed 3 minutes ago			
Creación de la clase Soldado con atributos nombre, vida, fila y columna. Incluye constructor, getters, setters y método toString para mostrar la información del soldado.		22bdc81	<>
SrEstrada committed 4 minutes ago			
Aquí esta mi main el cual esta llamando a los diferentes metodos de las clases Soldado y Viodejuego2		08c1303	<>
SrEstrada committed 7 minutes ago			
Add files via upload		Verified 8b33e8f	<>
SrEstrada authored 16 hours ago			
Update Soldado.java		Verified cce23ca	<>
SrEstrada authored 16 hours ago			
Add files via upload		Verified a648f48	<>
SrEstrada authored 16 hours ago			
Commits on Oct 15, 2024			
Add files via upload		Verified 77ea0ef	<>
SrEstrada authored 3 days ago			
Add files via upload		Verified a576afd	<>
SrEstrada authored 4 days ago			
Add files via upload		Verified 91f5780	<>
SrEstrada authored 4 days ago			

¿Con que valores comprobaste que tu práctica estuviera correcta?

Para validar la práctica, se utilizaron valores aleatorios generados por el programa, como la vida de los soldados (entre 1 y 5) y su posición en el tablero (en un tablero de 10x10). Las pruebas se realizaron ejecutando el programa varias veces para asegurar que los resultados fueran consistentes.

¿Qué resultado esperabas obtener para cada valor de entrada?

Tablero: Esperaba ver un tablero inicial con soldados posicionados aleatoriamente y celdas vacías bien representadas.



Ordenamiento (Burbuja e Inserción Binaria): El resultado esperado era que ambos algoritmos ordenaran a los soldados de mayor a menor nivel de vida.

¿Qué valor o comportamiento obtuviste para cada valor de entrada?

Tablero Inicial: Los soldados se generaron y colocaron en posiciones aleatorias, con las celdas vacías claramente mostradas como _____.

Ranking después de Burbuja: El ranking mostró a los soldados con más vida primero, como se esperaba.

Ranking después de Inserción Binaria: El orden fue el mismo que con Burbuja, demostrando que ambos algoritmos funcionaron correctamente.

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

III. CUESTIONARIO

CONCLUSIONES

El trabajo hecho demostró el correcto funcionamiento de ambos métodos de ordenamiento. La inserción binaria, aunque más eficiente para listas parcialmente ordenadas, no mostró una gran diferencia en esta implementación debido al tamaño relativamente pequeño de la lista de soldados. La generación aleatoria de posiciones y niveles de vida permitió validar la robustez del código bajo diferentes condiciones. Además, los resultados fueron consistentes en todas las ejecuciones.

METODOLOGÍA DE TRABAJO

Visualización inicial: Primero, me aseguré de visualizar cómo debía verse el código completo y estructurar las interacciones clave entre el tablero y los soldados.

Generación de datos: Planifiqué la forma en que los soldados serían generados aleatoriamente, tanto en su vida como en su posición.



Impresión del tablero: Decidí cómo debía imprimirse el tablero, asegurando que las celdas vacías y ocupadas se representaran claramente.

Implementación de algoritmos: Implementé los algoritmos de ordenamiento asegurándome de comparar correctamente el nivel de vida de los soldados.

Validación mediante pruebas: Ejecuté el código varias veces, asegurándome de que ambos algoritmos de ordenación proporcionarían los mismos resultados consistentes para diferentes ejecuciones.

REFERENCIAS Y BIBLIOGRAFÍA

GitHub - SrEstrada/Laboratorios_Estrada_Arce. (s.f.). GitHub.
https://github.com/SrEstrada/Laboratorios_Estrada_Arce.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: 11

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	3	
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		18	