

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p><b>Formato:</b> 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:	<i>Fundamentos de la Programación 2</i>				
TÍTULO DE LA PRÁCTICA:	<i>Arraylist</i>				
NÚMERO DE PRÁCTICA:	<i>06</i>	AÑO LECTIVO:	<i>2024 B</i>	NRO. SEMESTRE:	<i>II</i>
FECHA DE PRESENTACIÓN	<i>25/10/2024</i>	HORA DE PRESENTACIÓN	<i>18:20:00</i>		
INTEGRANTE (s) <i>Layme Salas Rodrigo Fabricio</i>				NOTA (0-20)	
DOCENTE(s): <i>Ing. Lino Jose Pinto Oppe</i>					

RESULTADOS Y PRUEBAS
<p><b>I. EJERCICIOS RESUELTOS:</b></p> <p style="text-align: center;"><b><i>CLASE VideoJuego3</i></b></p> <p style="text-align: center;"><b><i>(Las explicaciones están en los comentarios dentro del código)</i></b></p>

```

1  /*Propósito: Simular un tablero 10x10 en el que se desarrolla una batalla*/
2  import java.util.*;
3  public class VideoJuego3 {
4      public static int vidaTotalAzul = 0; //USO VARIABLES GLOBALES PARA HACER VARIOS CAMBIOS DESDE MÉTODOS
5      public static int vidaTotalRojo = 0;
6      public static double promedioAzul = 0;
7      public static double promedioRojo = 0;
8      public static Soldado mayorVidaAzul = new Soldado(nombre:null, vida:0, fila:0, columna:0, equipo:null); // INICIALIZO UN SOLDADO PARA LA
// COMPARACIÓN
9      public static Soldado mayorVidaRojo = new Soldado(nombre:null, vida:0, fila:0, columna:0, equipo:null);
10     public static Soldado[] soldadosUniDimensionalAzul = new Soldado[10]; // PARA NO DESPERDICAR TANTA MEMORIA ME APOYO DE UNA MATRIZ
11     public static Soldado[] soldadosUniDimensionalRojo = new Soldado[10]; // LAS USARÉ SOLO PARA ORDENAMIENTOS
Run | Debug
12     public static void main(String[] args) {
13         ArrayList<ArrayList<Soldado>> tablero = new ArrayList<ArrayList<Soldado>>(); //INICIALIZO MI ARRAYLIST BIDIMENSIONAL
14         int cantidad = (int)(Math.random() * 10 + 1);
15         int cantidadEnemiga = (int)(Math.random() * 10 + 1);
16         for(int i = 0; i<10; i++){
17             tablero.add(new ArrayList<Soldado>());
18             for(int j = 0; j<10;j++){
19                 tablero.get(i).add(new Soldado(nombre:null, vida:0, fila:0, columna:0, equipo:null)); // INICIALIZA CON VALORES NULL Y 0
20             }
21             inicializarEjercito(tablero, cantidad); // METO DATOS A MI ARRAYLIST
22             inicializarEjercitoEnemigo(tablero, cantidadEnemiga);
23             mostrarTabla(tablero); // MUESTRA TABLA
24             hallarSoldadoMayorVida(tablero);
25             System.out.println("El soldado con mayor vida del ejército azul es: " + mayorVidaAzul);
26             System.out.println("El soldado con mayor vida del ejército rojo es: " + mayorVidaRojo);
27             hallarPromedioVida(tablero, cantidad, cantidadEnemiga);
28             System.out.println("El promedio del ejército azul es: " + promedioAzul);
29             System.out.println("El promedio del ejército rojo es: " + promedioRojo);
30             System.out.println(x:"DATOS DEL EJÉRCITO AZUL POR ORDEN DE INGRESO:");
31             imprimirInformacion(cantidad, soldadosUniDimensionalAzul); // IMPRIME LA INFORMACIÓN DE LOS SOLDADOS DE UN EJÉRCITO
32             System.out.println(x:"DATOS DEL EJÉRCITO ROJO POR ORDEN DE INGRESO:");
33             imprimirInformacion(cantidadEnemiga, soldadosUniDimensionalRojo);
34             rankingDePoder(cantidad, soldadosUniDimensionalAzul); // ORDENA POR MÉTODO BURBUJA
35             ordenarSeleccion(cantidadEnemiga, soldadosUniDimensionalRojo); // ORDENA POR MÉTODO SELECCIÓN
36             System.out.println(x:"DATOS DEL EJÉRCITO AZUL ORDENADO POR NIVEL DE VIDA:");
37             imprimirInformacion(cantidad, soldadosUniDimensionalAzul);
38             System.out.println(x:"DATOS DEL EJÉRCITO ROJO ORDENADO POR NIVEL DE VIDA:");
39             imprimirInformacion(cantidadEnemiga, soldadosUniDimensionalRojo);
40             mostrarGanador(); // MUESTRA EL GANADOR, CRITERIO: VIDA TOTAL DE LOS EJÉRCITOS

```

```

41     }
42     public static void inicializarEjercito(ArrayList<ArrayList<Soldado>> tablero, int cantidad) { // INICIALIZA SOLDADOS CON SU INFORMACIÓN
43         int contadorNombreSoldado = 0;
44         while (cantidad > 0) {
45             int fila = (int) (Math.random() * 10);
46             int columna = (int) (Math.random() * 10);
47             if (tablero.get(fila).get(columna).getVida() == 0) { // SE VERIFICA QUE NO HAYA OTRO SOLDADO EN ESA POSICIÓN, SI LO HAY, BUSCA OTRA
48                 tablero.get(fila).set(columna, new Soldado("\u001B[44mSoldado"+fila+"X"+columna + "\u001B[0m", (int) (Math.random() * 5 + 1), fila,
49                     columna, equipo:"azul")); //USÉ COLORES PARA LA DISTINCIÓN ENTRE SOLDADOS
50                 soldadosUniDimensionalAzul[contadorNombreSoldado] = tablero.get(fila).get(columna);
51                 contadorNombreSoldado++;
52                 cantidad--;
53                 vidaTotalAzul += tablero.get(fila).get(columna).getVida();
54             }
55         }
56     public static void inicializarEjercitoEnemigo(ArrayList<ArrayList<Soldado>> tablero, int cantidad) { // INICIALIZA SOLDADOS CON SU INFORMACIÓN
57         int contadorNombreSoldado = 0;
58         while (cantidad > 0) {
59             int fila = (int) (Math.random() * 10);
60             int columna = (int) (Math.random() * 10);
61             if (tablero.get(fila).get(columna).getVida() == 0) { // SE VERIFICA QUE NO HAYA OTRO SOLDADO EN ESA POSICIÓN, SI LO HAY, BUSCA OTRA
62                 tablero.get(fila).set(columna, new Soldado("\u001B[41mSoldado"+fila+"X"+columna + "\u001B[0m", (int) (Math.random() * 5 + 1), fila,
63                     columna, equipo:"rojo")); //USÉ COLORES PARA LA DISTINCIÓN ENTRE SOLDADOS
64                 soldadosUniDimensionalRojo[contadorNombreSoldado] = tablero.get(fila).get(columna);
65                 contadorNombreSoldado++;
66                 cantidad--;
67                 vidaTotalRojo += tablero.get(fila).get(columna).getVida();
68             }
69         }
70     public static void mostrarTabla(ArrayList<ArrayList<Soldado>> tablero) { // GENERA A BASE DE | Y _ UN TABLA PARA LOS SOLDADOS
71         for (int i = 0; i < tablero.size(); i++) {
72             for (int j = 0; j < tablero.get(i).size(); j++) {
73                 if (tablero.get(i).get(j).getVida() == 0) // GENERA ESPACIOS VACÍOS SI NO HAY UN OBJETO EN ESA DIRECCIÓN
74                     System.out.print(s:"|_____");
75                 else
76                     System.out.print(s:"|" + tablero.get(i).get(j).getNombre()); // CARGA EL NOMBRE DEL SOLDADO SI EXISTE
77             }
78             System.out.println(x:"|"); // ACOMODA EL TABLERO 10X10

```

```

79     }
80 }
81 public static void hallarSoldadoMayorVida(ArrayList<ArrayList<Soldado>> tablero) { // MUESTRA AL SOLDADO CON MAYOR VIDA
82     for (int i = 0; i < tablero.size(); i++) {
83         for (int j = 0; j < tablero.get(i).size(); j++) {
84             if (tablero.get(i).get(j).getVida() != 0 && tablero.get(i).get(j).getEquipo().equals(anObject:"azul") && mayorVidaAzul.getVida() <
85                 tablero.get(i).get(j).getVida()) // SOLO SE CUMPLE SI LA VIDA ES DIFERENTES DE 0, PERTENECE AL EQUIPO AZUL Y ES MAYOR
86                 mayorVidaAzul = tablero.get(i).get(j);
87             if (tablero.get(i).get(j).getVida() != 0 && tablero.get(i).get(j).getEquipo().equals(anObject:"rojo") && mayorVidaRojo.getVida() <
88                 tablero.get(i).get(j).getVida()) // SOLO SE CUMPLE SI LA VIDA ES DIFERENTES DE 0, PERTENECE AL EQUIPO ROJO Y ES MAYOR
89                 mayorVidaRojo = tablero.get(i).get(j);
90         }
91     }
92     public static void hallarPromedioVida(ArrayList<ArrayList<Soldado>> tablero, int cantidad, int cantidadEnemiga) { //MUESTRA EL PROMEDIO DE VIDA
93     TRBAJA EN LA MATRIZ BIDIMENSIONAL
94     for (int i = 0; i < tablero.size(); i++) {
95         for (int j = 0; j < tablero.get(i).size(); j++) {
96             if (tablero.get(i).get(j).getVida() != 0 && tablero.get(i).get(j).getEquipo().equals(anObject:"azul"))
97                 promedioAzul += tablero.get(i).get(j).getVida(); // AUMENTA LA VARIABLE GLOBAL
98             if (tablero.get(i).get(j).getVida() != 0 && tablero.get(i).get(j).getEquipo().equals(anObject:"azul"))
99                 promedioRojo += tablero.get(i).get(j).getVida();
100         }
101     }
102     promedioAzul /= cantidad;
103     promedioRojo /= cantidadEnemiga; // CONVIERTE LA VARIABLE GLOBAL EN PROMEDIO EN LUGAR DE UNA SUMATORIA
104 }
105 public static void rankingDePoder(int cantidad, Soldado[] soldadosUniDimensional) { //PRIMER ALGORITMO DE ORDENAMIENTO (BURBUJA)
106     boolean intercambio = true;
107     while (intercambio) {
108         intercambio = false; // LO MANTIENE FALSO HASTA QUE SE HAYA UN INTERCAMBIO, SINO SE SALE DEL BUCLE
109         for (int i = 0; i < cantidad - 1; i++)
110             if (soldadosUniDimensional[i].getVida() < soldadosUniDimensional[i + 1].getVida()) {
111                 intercambio = true;
112                 Soldado temp = new Soldado(nombre:null, vida:0, fila:0, columna:0, equipo:null); //VARIABLE TEMPORAL PARA EL INTERCAMBIO
113                 temp = soldadosUniDimensional[i + 1];
114                 soldadosUniDimensional[i + 1] = soldadosUniDimensional[i];
115                 soldadosUniDimensional[i] = temp;
116             }
117     }

```

```

116     }
117 }
118 public static void ordenarSeleccion(int cantidad, Soldado[] soldadosUniDimensional) { // 2DO ALGORITMO DE ORDENAMIENTO
119     for (int i = 0; i < cantidad - 1; i++) { //SE USA LA CANTIDAD PARA EVITAR QUE EL BUCLE SEÑALE A OBJETOS NULL
120         int menor = i;
121         for (int j = i + 1; j < cantidad; j++)
122             if (soldadosUniDimensional[j].getVida() > soldadosUniDimensional[menor].getVida())
123                 menor = j; // Almacena la posición del menor
124         Soldado temp = soldadosUniDimensional[menor]; // Intercambio de elementos
125         soldadosUniDimensional[menor] = soldadosUniDimensional[i];
126         soldadosUniDimensional[i] = temp;
127     }
128 }
129 public static void imprimirInformacion(int cantidad, Soldado[] soldadosUniDimensional){
130     for(int i = 0; i<cantidad; i++){
131         System.out.println("SOLDADO " + i + ":");
132         System.out.println(soldadosUniDimensional[i].toString()); //ME APOYO DE UNA MATRIZ UNIDIMENSIONAL PARA NO
133         // DESPERDICIA MEMORIA
134     }
135 }
136 public static void mostrarGanador(){ // CRITERIO: CANTIDAD TOTAL DE VIDA
137     if(vidaTotalAzul>vidaTotalRojo)
138         System.out.println("¡El ejercito azul gana por mayor nivel de vida! " + "\nAzul " + vidaTotalAzul + ":" + vidaTotalRojo + " Rojo");
139     else if(vidaTotalAzul<vidaTotalRojo)
140         System.out.println("¡El ejercito rojo gana por mayor nivel de vida! " + "\nAzul " + vidaTotalAzul + ":" + vidaTotalRojo + " Rojo");
141     else
142         System.out.println("¡Ha ocurrido un empate! " + "\nAzul " + vidaTotalAzul + ":" + vidaTotalRojo + " Rojo");
143 }

```

## CLASE Soldado

*(Las explicaciones están dentro de los comentarios)*

```

1 public class Soldado {
2     private String nombre;
3     private int vida;
4     private int fila;
5     private int columna;
6     private String equipo;
7     public Soldado(String nombre, int vida, int fila, int columna, String equipo){ // CONSTRUCTOR PARA EVITAR MUCHOS MÉTODOS
8         this.nombre = nombre;
9         this.vida = vida;
10        this.fila = fila;
11        this.columna = columna;
12        this.equipo = equipo;
13    }
14    public String getNombre() {
15        return nombre;
16    }
17    public int getVida() {
18        return vida;
19    }
20    public String getEquipo() {
21        return equipo;
22    }
23    public String toString() {
24        return "Información:\nNombre: " + nombre + "\nVida: " + vida + "\nFila: " + fila + "\nColumna: " + columna;
25    }
26 }

```

## PRUEBAS

									Soldado0X9
		Soldado1X2							
		Soldado2X2		Soldado2X4					Soldado2X9
			Soldado3X3						
		Soldado4X2		Soldado4X4					
			Soldado6X3						
Soldado7X0									
	Soldado8X1								
Soldado9X0									

Así se vería el tablero inicializado, usé colores para diferenciar los soldados de los ejércitos

```

El soldado con mayor vida del ejército azul es: Información:
Nombre: Soldado0X9
Vida: 5
Fila: 0
Columna: 9
El soldado con mayor vida del ejército rojo es: Información:
Nombre: Soldado2X4
Vida: 5
Fila: 2
Columna: 4
El promedio del ejército azul es: 4.0
El promedio del ejército rojo es: 8.0

```

*Aquí se calculan los soldados con mayor vida y se halla el promedio de vida de cada ejército.*

<p>DATOS DEL EJÉRCITO AZUL POR ORDEN DE INGRESO:</p> <p>SOLDADO 0: Información: Nombre: Soldado4X4 Vida: 4 Fila: 4 Columna: 4</p> <p>SOLDADO 1: Información: Nombre: Soldado8X1 Vida: 3 Fila: 8 Columna: 1</p> <p>SOLDADO 2: Información: Nombre: Soldado0X9 Vida: 5 Fila: 0 Columna: 9</p> <p>SOLDADO 3: Información: Nombre: Soldado6X3 Vida: 5 Fila: 6 Columna: 3</p> <p>SOLDADO 4: Información:</p>	<p>Nombre: Soldado2X9 Vida: 5 Fila: 2 Columna: 9</p> <p>SOLDADO 5: Información: Nombre: Soldado7X0 Vida: 2 Fila: 7 Columna: 0</p> <p>SOLDADO 6: Información: Nombre: Soldado4X2 Vida: 4 Fila: 4 Columna: 2</p> <p>SOLDADO 7: Información: Nombre: Soldado9X0 Vida: 4 Fila: 9 Columna: 0</p>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

*Así se vería los soldados del ejército azul por orden de ingreso*

DATOS DEL EJÉRCITO ROJO POR ORDEN DE INGRESO:

SOLDADO 0:  
Información:  
Nombre: Soldado3X3  
Vida: 5  
Fila: 3  
Columna: 3  
SOLDADO 1:  
Información:  
Nombre: Soldado1X2  
Vida: 4  
Fila: 1  
Columna: 2  
SOLDADO 2:  
Información:  
Nombre: Soldado2X4  
Vida: 5  
Fila: 2  
Columna: 4  
SOLDADO 3:  
Información:  
Nombre: Soldado2X2  
Vida: 4  
Fila: 2  
Columna: 2

Así se vería los soldados del ejército rojo por orden de ingreso

DATOS DEL EJÉRCITO AZUL ORDENADO POR NIVEL DE VIDA:

SOLDADO 0:  
Información:  
Nombre: Soldado0X0  
Vida: 5  
Fila: 0  
Columna: 9  
SOLDADO 1:  
Información:  
Nombre: Soldado6X3  
Vida: 5  
Fila: 6  
Columna: 3  
SOLDADO 2:  
Información:  
Nombre: Soldado2X9  
Vida: 5  
Fila: 2  
Columna: 9  
SOLDADO 3:  
Información:  
Nombre: Soldado4X4  
Vida: 4  
Fila: 4  
Columna: 4  
SOLDADO 4:  
Información:  
Nombre: Soldado4X2  
Vida: 4  
Fila: 4

Columna: 2  
SOLDADO 5:  
Información:  
Nombre: Soldado9X0  
Vida: 4  
Fila: 9  
Columna: 0  
SOLDADO 6:  
Información:  
Nombre: Soldado8X1  
Vida: 3  
Fila: 8  
Columna: 1  
SOLDADO 7:  
Información:  
Nombre: Soldado7X0  
Vida: 2  
Fila: 7  
Columna: 0

Así se verían los soldados azules ordenados descendientemente por nivel de vida. (Método de burbuja)

```

DATOS DEL EJÉRCITO ROJO ORDENADO POR NIVEL DE VIDA:
SOLDADO 0:
Información:
Nombre: Soldado3X3
Vida: 5
Fila: 3
Columna: 3
SOLDADO 1:
Información:
Nombre: Soldado2X4
Vida: 5
Fila: 2
Columna: 4
SOLDADO 2:
Información:
Nombre: Soldado1X2
Vida: 4
Fila: 1
Columna: 2
SOLDADO 3:
Información:
Nombre: Soldado2X2
Vida: 4
Fila: 2
Columna: 2

```

Así se verían los soldados rojos ordenados descendentemente por nivel de vida. (Método de selección)

```

¡El ejercito azul gana por mayor nivel de vida!
Azul 32:18 Rojo

```

El criterio del ganador es de acuerdo a quien tiene mayor cantidad de puntos de vida. **¡ESO ES TODO!**

## II. PRUEBAS

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

*Con valores int y String, además datos que parecía que el programa aceptaría, como caracteres especiales para probar como funciona cada método.*



*¿Qué resultado esperabas obtener para cada valor de entrada?*

*Esperaba que se almacenara dentro del objeto y atributo que quería; esperaba no tener errores, pero tuve varios al momento de construir un ArrayList con valores nulos, pronto lo resolví con otro enfoque.*

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

*Obtuve los espera, una secuencia limpia de los métodos usados en el main y sin ningún error.*



	<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: 9</p>

### III. CUESTIONARIO:

#### PRUEBAS DE COMMIT HECHO EN GIT BASH:

**LINK A MI REPOSITORIO DE GIT HUB:** <https://github.com/F4brici0L4yme/PF2.git>

*Hice un git status para ver qué archivos tenía sin trackear.*

```
PS D:\FP2 LABORATORIO\PF2\LAYME_SALAS_LABORATORIO_06> git status
On branch main
Your branch is up to date with 'origin/main'.
```

*Había modificado mis 2 archivos de java y los añadí con un git add .*


```
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Soldado.java
        modified:   VideoJuego3.java

no changes added to commit (use "git add" and/or "git commit -a")
PS D:\FP2 LABORATORIO\PF2\LAYME_SALAS_LABORATORIO_06> git add .
warning: in the working copy of 'LAYME_SALAS_LABORATORIO_06/Soldado.java', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'LAYME_SALAS_LABORATORIO_06/VideoJuego3.java', LF will be replaced by CRLF the next time Git touches it
```

*Finalmente, usé el comando git push para mandar todos los cambios a la nube*

```
PS D:\FP2 LABORATORIO\PF2\LAYME_SALAS_LABORATORIO_06> git commit -m "LABORATORIO 6 TERMINADO"
[main 8a64237] LABORATORIO 6 TERMINADO
 2 files changed, 91 insertions(+), 37 deletions(-)
PS D:\FP2 LABORATORIO\PF2\LAYME_SALAS_LABORATORIO_06> git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 16 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 2.25 KiB | 1.12 MiB/s, done.
Total 5 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/F4brici0L4yme/PF2.git
 d4ba368..8a64237  main -> main
PS D:\FP2 LABORATORIO\PF2\LAYME_SALAS_LABORATORIO_06> |
```

*Ya están reflejados y pueden ser vistos en GitHub.*

 LAYME_SALAS_LABORATORIO_06	LABORATORIO 6 TERMINADO	2 minutes ago
----------------------------------------------------------------------------------------------------------------	-------------------------	---------------

	<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		
<b>Aprobación:</b> 2022/03/01	<b>Código:</b> GUIA-PRLE-001	<b>Página:</b> 10

## CONCLUSIONES

*Fue interesante cambiar arrays por arraylist en el mismo trabajo, fue como una migración de tecnología. Siento que los ArrayList son mucho más versátiles que los arrays, pero cada uno tiene su magia.*

## METODOLOGÍA DE TRABAJO

*Usé las mismas que fui usando durante estos laboratorios, comentar bloques de código para poder concentrarme en una parte y revisando problemas pasados y similares que ya resolví para tener una idea y construir un nuevo método.*

## REFERENCIAS Y BIBLIOGRAFÍA

*E. G. Castro Gutiérrez y M. W. Aedo López, Fundamentos de programación 2: tópicos de programación orientada a objetos, 1st ed. Arequipa, Perú: Universidad Nacional de San Agustín, 2021, pp. 170, ISBN 978-612-5035-20-2.*

**RÚBRICA DE CALIFICACIÓN DE LABORATORIO**

**(EN LA SIGUIENTE PÁGINA)**

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	2	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	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	

	<p align="center"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<p align="center"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLE-001</p>	<p><b>Página:</b> 12</p>

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		19	